Covert Channel Communication Using Protocol Field Manipulation

Overview

This project implements a **covert storage channel** that exploits **protocol field manipulation** using the **TTL (Time-to-Live)** field in the IP header. The covert channel allows for secret communication between a sender and a receiver by embedding data into specific fields of network packets.

The sender encodes information into the TTL field, and the receiver decodes it based on an agreed-upon consensus.

Implementation Details

Encoding Scheme

The TTL field is divided into **ranges** to encode 2 bits per packet (can be changed in the config.json):

- The range of TTL values is divided into `num_ttl_ranges` equal-sized ranges.
- Each range represents one of the four possible 2-bit combinations (`00` , `01` , `10` , `11`).
- The sender randomly selects a TTL value within the appropriate range for each 2-bit chunk of the binary message.

Decoding Scheme

The receiver decodes the message by:

1. Extracting the TTL value from each received packet.
2. Determining which range the TTL value belongs to.
3. Mapping the range to the corresponding 2-bit combination.
4. Reconstructing characters from 8-bit chunks.

Parameters

The implementation is fully parametric, and all values are defined in `config.json` :

Sender Parameters

- `log_file_name` : File name for logging sent and received messages.
- `min_ttl_val` : Minimum TTL value (e.g., 32).
- `max_ttl_val` : Maximum TTL value (e.g., 224).
- `num_ttl_ranges` : Number of TTL ranges (e.g., 4 for 2-bit encoding).

Receiver Parameters

- `log_file_name` : File name for logging sent and received messages.
- `min_ttl_val` : Minimum TTL value (e.g., 32).
- `max_ttl_val` : Maximum TTL value (e.g., 224).
- `num_ttl_ranges` : Number of TTL ranges (e.g., 4 for 2-bit encoding).
- `sender_ip` : IP address of the sender

Limitations

1. **TTL Range**: The TTL field must remain within valid values (1–255). The selected ranges (`min_ttl_val` to `max_ttl_val`) must not overlap with reserved or invalid ranges.
2. **Network Delays**: Packet delays may affect decoding accuracy if packets are dropped or reordered.
3. **Stealth**: While we added randomness to TTL values, excessive use of certain ranges may raise suspicion in monitored networks.

Measuring Covert Channel Capacity

The covert channel capacity was measured using the following steps:

1. A binary message of length 128 bits (16 characters) was generated.
2. The timer started just before sending the first packet.
3. The timer stopped after sending the last packet.
4. The time difference (in seconds) was calculated.
5. Capacity was calculated as:

$$\text{Capacity} = \frac{128}{\text{Time (seconds)}}$$

Results

- **Measured Capacity**: Our covert channel achieved a capacity of approximately **129.02 bits per second** with 2 bits encoded per packet

How It Works

Sender (`send` Function)

1. Generates a random binary message using `generate_random_binary_message_with_logging`.
2. Divides the binary message into 2-bit chunks.
3. Selects a random TTL value within the appropriate range for each chunk.
4. Sends packets with encoded TTL values using UDP.

Receiver (`receive` Function)

1. Captures incoming UDP packets using Scapy's `sniff`.
2. Extracts TTL values and maps them to their corresponding ranges.
3. Decodes 2-bit chunks from TTL values and reconstructs characters.
4. Stops decoding when it encounters the stopping character (`.`).

Usage Instructions

1. Clone this repository and set up the Docker environment as described in the project documentation.
2. Edit `config.json` to configure parameters for your setup
3. Run `make receive` in one terminal to start capturing packets on the receiver side.
4. Run `make send` in another terminal to send packets from the sender side.
5. Use `make compare` to verify that sent and received messages match.

```
{
  "covert_channel_code": "CSC-PSV-IP-TTL",
  "send": {
    "parameters": {
      "log_file_name": "CovertChannelSender.log",
      "min_ttl_val": 32,
      "max_ttl_value": 224,
      "num_ttl_ranges": 4
    }
  },
  "receive": {
    "parameters": {
      "log_file_name": "CovertChannelReceiver.log",
      "min_ttl_val": 32,
      "max_ttl_value": 224,
      "num_ttl_ranges": 4,
      "sender_ip": "172.18.0.2"
    }
  }
}
```


Design Considerations

Maximizing Capacity

- By encoding 2 bits per packet, we maximize capacity while maintaining stealth. It can be increased using the `num_ttl_ranges` parameter.
- Randomized TTL values within each range reduce detectability.

Stealth

- Randomness in TTL values ensures that traffic patterns remain normal.
- Using valid protocol fields minimizes suspicion from network monitoring tools.

Known Issues

1. **Packet Loss**: If packets are dropped due to network congestion, decoding errors may occur.
2. **Reordering**: Out-of-order packets can disrupt message reconstruction.

Conclusion

Our implementation successfully demonstrates a covert storage channel using protocol field manipulation.